

.....	TOPMEM (pointer as in interpreter).....
DYNAMIC STRINGS	
.....	TOPCODE (pointer at 2884/2085).....
CODE	
.....	STARTCODE (pointer at 2082/2083).....
INITIALISATION DATA	
.....	STARTINT (pointer at 2080/2081).....
ARRAYS	
.....	ARRAYTAB (pointer at 2078/2079).....
DATA STATEMENTS	
.....	STARTDATA (pointer at 2076/2077).....
VARIABLES	
.....	10240
SPEEDCODE INTERPRETER	
.....	10 SYS (2073) PETSPEED
.....	2048
PAGE ZERO, STACK, TAPE BUFFERS, SCREEN ETC	
.....	0

The Speedcode version of a Basic program is usually between 0.5 and 0.65 the size of the original Basic text. To this, however, must be added the 8k overhead of the Speedcode interpreter.

Whereas a BASIC 2.0 program creates its variables and arrays when it is run, a Petspeed program does so at compile time. TOPCODE is the highest address used by the program except for dynamic strings which are built up from TOPMEM downwards.

Accessing Variables

The location of each variable and array can be found by running the supplied program REPORT on the Utilities disk. The program asks for the source file name and allows output to be directed to screen or printer. REPORT lists all variables, arrays and user defined functions and gives their addresses. In addition, the locations of the main program segments are supplied. The only address missing from the report is that of TOPCODE (see above) which may easily be determined after compilation.

The diagram below gives the format of all Petspeed data types. Corresponding diagrams illustrating the format of BASIC 2.0 variables are to be found in Commodore's own documentation.

A word of warning is necessary here. Variables and array elements which are not declared as integers using the % sign will not necessarily be held in floating point format. The type of the data in an ordinary real variable or array element may change frequently at run time. The way to transfer values from Petspeed variables to machine code routines is to make them integer explicitly (e.g. A%, PQ%(10,2) etc). If this is not possible, the machine code routine can if necessary obtain the current type of a variable by looking at the type descriptor (see below). Should the programmer wish to pass a value back to Petspeed via a real/int variable, he may do so by forcing the type descriptor to the value appropriate for the type of data being passed.

Simple Variables (Bytes marked X are not used)

Type	B1	B2	B3	B4	B5	B6	B7
Real/Integer	EXP	M1	M2/MSB	M3/LSB	M4	SGN	0/1
Declared Integer	X	X	MSB	LSB	X	X	5
String	X	X	LENGTH	PTR-LSB	PTR-MSB	X	-ve
Function	X	X	X	PTR-LSB	PTR-MSB	X	X

The current type of Real/Int variables is specified by B7. The 32 most frequently accessed variables are treated specially and contain an extra unused byte (B8).

Arrays

Type	B1	B2	B3	B4	B5
Real/Integer	EXP/0	M1 & SGN/x	M2/MSB	M3/LSB	M4/x
Declared Integer	MSB	LSB	(2 bytes used)		
String	LENGTH	PTR-LSB	PTR-MSB	(3 bytes used)	

The individual elements of Real/Int arrays can also change type. If the exponent is zero, the type is integer, otherwise it is real.

PET SPEED 64



Oxford Computer Systems (Software) Ltd.

Hensington Road, Woodstock, Oxford OX7 1JR Tel. (0993) 812700 Telex 83147 Ref. OCSL

Preparing Petspeed for use

The system disk supplied with this manual cannot itself be used to compile programs. Before Petspeed can be used it is necessary to create two Master disks, a Petspeed Master and a Utilities Master. When these have been created using the instructions below, they should be backed up (using a program supplied) and thereafter kept, along with the system disk, in a safe place. When further Petspeed or utility disks are required, they should be made from the Masters and not the system disk. The system disk should only be used in the event of the Masters becoming corrupt. To create the Masters, proceed as follows:

- 1 Format 4 disks and mark them PETSPEED MASTER, UTILITIES MASTER, PETSPEED WORK and UTILITIES WORK.
- 2 Place the system disk in the 1541 drive, LOAD "BACKUP", 8 and RUN.
- 3 From the 4 menu options displayed, select MAKE PETSPEED MASTER and follow the instructions given in the program.
- 4 When the menu reappears, select MAKE UTILITIES MASTER and again follow the instructions displayed. The system disk may now be put away.
- 5 Select BACKUP PETSPEED and follow the instructions to make a Petspeed work disk.
- 6 Select BACKUP UTILITIES and follow the instruction to make a utilities work disk.

The security podule

The security podule supplied with Petspeed should be plugged either into the cassette port or into control port 2 depending upon the type of podule supplied. The podule is only necessary during compilation.

RUNNING PETSPEED

- 1 Ensure that the podule is plugged into the 1st cassette or control port of the computer.
- 2 Save the program to be compiled on the Petspeed system disk.
- 3 Type LOAD "PETSPEED", 8 followed by RETURN and then RUN.
- 4 When requested, enter the source program name followed by RETURN. the compiler scans the source program four times and as it does so displays the numbers of the lines being processed.

When compilation is complete, the screen will clear and the statement

RUN: NNNNN bytes free

will appear at the top of the screen. The compiled program is now both in memory and on the disk in the 1541 drive. To run, simply type the HOME key followed by RETURN.

The compiled program on the work disk has the same name as the original source program but with the characters .WOW appended. It will LOAD and RUN just as a normal BASIC program.

Disk space on the 1541

During compilation Petspeed generates several work files. The largest of these, the parse tree, will often run into several hundred blocks and in some cases there will be insufficient space to accommodate it. Whenever the compiler is used the disk should contain only the Petspeed system files and the source program.

Facilities not available under Petspeed

- 1 RUN may be used but not with a linenumber as its parameter. LIST, CONT and SAVE are not allowed.
- 2 When a program is overlayed all variables are cleared.
- 3 Petspeed cannot cope with dynamic arrays. This means that each array may only be DIMensioned once in a given program and, whenever a statement like:

```
100 DIM A(N)
```

occurs in a program, it should be changed so that the compiler knows at compile time how much store to reserve. For example, line 100 above might be changed to:

```
100 DIM A(50)
```

If this is not done, the compiler will stop during pass 1 and ask for the information, e.g:

```
SUBSCRIPT 0 FOR A(IN LINE 100 ?
```

When the information is given, compilation will continue.

Because of the overhead of 35 blocks built into all compiled programs, those which occupy less than about 70 blocks of disk space may become larger when compiled. Programs larger than 70 blocks will usually compile smaller. Because Petspeed saves variables and arrays along with the program, programs will appear larger on the disk than they actually are.

Extra programming facilities

- 1 User defined string and mixed functions are allowed. Either the function itself or its argument or both may be of either string or numeric type, e.g. DEF FN A(X), DEF FN A\$(X), DEF FN A(X\$) and DEF FN A\$(X\$) are all permissible.
- 2 Integer FOR loops such as the following are allowed:
FOR I%=0 TO 10::NEXT I%
- 3 The compiler directive REM!LN, included at the beginning of a program will have the effect of making all characters in a variable name significant.
Petspeed defaults to the same treatment of variable names as BASIC 2.0 (two significant characters) and the above directive should only be used where it is specifically required.
- 4 Two statements REM!ES and REM!DS are provided to enable and disable the STOP key respectively. The STOP key is disabled by default.

ERRORS

All the usual BASIC 2.0 errors may occur during compilation and most of these will cause compilation to abort. The only type of compile time error which will not cause the compiler to abort corresponds to BASIC 2.0's UNDEF'D STATEMENT ERROR. Such errors will be found during pass 4 and, in these circumstances, the linenumbers at which the errors occur will appear in reverse field. A program containing references to undefined statements will run, provided that the statements containing such references are never executed. If this should happen the run will terminate with message:

UNDEF'D STATEMENT ERROR IN XXXXX

An error that may occur at compile time where it would not have done in BASIC 2.0 is FORMULA TOO COMPLEX ERROR. This may happen if the 3rd argument of a MID\$ contains function or array references. The problem may be easily solved by replacing the 3rd argument with a variable.

It should be noted that occasionally the line number given in an error report is off by one line, that is, the error actually occurred in the line preceding the one reported.

When compilation is aborted due to a compile time error, certain work files are left on the disk. In order to conserve disk space these may have to be scratched prior to the next compilation.

Run Time Errors

With the exception of BAD SUBSCRIPT ERROR, Petspeed can produce all the BASIC 2.0 error messages at run time.

All run time errors except UNDEF'D STATEMENT ERROR give a code address instead of a linenumber. In order to locate the corresponding line, it is necessary to use the error locating program ERRORS, previously copied to the utilities disk. ERRORS accesses both the .WOW file and the .W file which should both be present on the disk in the 1541. When the program is run, it asks for the name of the program in which the error occurred and the address given in the error report. After a few seconds the associated linenumber is displayed.

Interfacing machine code routines to Petspeed

Petspeed expects a clean Basic program as a source file. If the program to be compiled contains machine code routines, these should be removed before compilation.

Machine language programs which access BASIC 2.0 variables will not work without alteration as Petspeed holds its variables and arrays differently. The following notes provide all the information required for making the necessary changes.

Object programs generated by Petspeed consist of a p-code (Speedcode) optimised version of the source program and a run time interpreter. The full memory map is as follows: